Developing Safety-Critical Systems with UML

Jan Jürjens
Software & Systems Engineering
TU Munich, Germany
juerjens@in.tum.de
http://www.jurjens.de/jan

Abstract. Safety-critical systems have to be developed carefully to prevent loss of life and resources due to system failures. Some of their mechanisms (for example, providing fault-tolerance) can be complicated to design and use correctly in the system context and are thus error-prone. We show how one can use UML for model-based development of safety-critical systems with the aim to increase the quality of the developed systems without an unacceptable increase in cost and time-to-market. Specifically, we describe how to use the UML extension mechanisms to include safety-requirements in a UML model which is then analyzed for satisfaction of the requirements. The approach can thus be used to encapsulate safety engineering knowledge. It is supported by a prototypical XMI-based tool performing the analysis.

1 Introduction

There is an increasing desire to exploit the flexibility of software-based systems in the context of critical systems where predictability is essential. Examples include the use of embedded systems in various application domains, such as fly-by-wire in Avionics, drive-by-wire in Automotive and so on. Given the high safety requirements in such systems (such as a maximum of $10^{-9}$ failures per hour in the avionics sector), a thorough design method is necessary. In particular, the use of redundancy mechanisms to compensate the faults that occur in any operational system may require complex protocols whose correctness can be non-obvious [Rus94]. Therefore, safety mechanisms cannot be “blindly” inserted into a critical system, but the overall system development must take safety aspects into account. Furthermore, sometimes safety mechanisms cannot be used off-the-shelf, but have to be designed specifically to satisfy given requirements. This can be non-trivial, as spectacular examples for software failures in practice demonstrate (such as the explosive failure of the Ariane 5 rocket in 1997).

Any support to aid safe systems development would thus be useful. In particular, it would be desirable to consider safety aspects already in the design phase, before a system is actually implemented, since removing flaws in the design phase saves cost and time. This is significant; for example, in avionics, verification costs represent 50% of the overall costs [Ran00]. There has been a significant amount of successful research into using formal methods for the
development of safety-critical systems. Unfortunately, part of the difficulty of critical systems development is that correctness is often in conflict to cost. It would thus be beneficial to use rigorous means in the context of an industrially efficient development method.

The Unified Modeling Language (UML) [UML01] offers an unprecedented opportunity for high-quality critical systems development that is feasible in an industrial context.

– As the de facto standard in industrial modeling, a large number of developers is trained in UML, making less training necessary. Also, UML specifications may already be available for safety analysis, which again would save time and cost.

– Compared to previous notations with a user community of comparable size, UML is relatively precisely defined, opening up the possibility for advanced tool-support to assist the development of safety-critical systems.

Problems in critical systems development often arise when the conceptual independence of software from the underlying physical layer turns out to be an unfaithful abstraction (for example in settings such as real-time or more generally safety-critical systems, see [Sel02]). Since UML allows the modeller to describe different views on a system, including the physical layer, it seems promising to try to use UML to address these problems by modeling the interdependencies between the system and its physical environment.

While there has been a significant amount of work addressing real-time systems with UML (including for example [SR98]), and increasing attention to using UML for security (see for example [Jü03c]), in the present work we consider safety and fault-tolerance requirements. To support safe systems development, safety checklists have been proposed in [HIL96,Lut96,Hel98]. Here, we tailor UML to this application domain by precisely defining some such checks with stereotypes capturing safety requirements and related physical properties. This way we encapsulate knowledge on prudent safety engineering and thereby make it available to developers who may not be specialized in safety. One can also check whether the constraints associated with the stereotypes are fulfilled in a given specification. A prototypical tool supporting this will shortly be introduced at the end of this paper.

Safety. In safety-critical systems, an important concept also used here is that of a safety level (see e.g. [Ran00]). Safety goals for safety-critical systems are often expressed quantitatively via the maximum allowed failure rate. We exemplarily consider the following kinds of failure semantics in this paper (other kinds have to be omitted for space reasons).

– crash/performance failure semantics means that a component may crash or may deliver the requested data only after the specified time limit, but it is assumed to be partially correct.

– value failure semantics means that a component may deliver incorrect values.